导航

博客园
首 页
新随笔
联 系
订 阅 XML
管 理

SSM框架——Spring+SpringMVC+Mybatis的搭建教程

一：概述

SSM框架在项目开发中经常使用到，相比于SSH框架，它在仅几年的开发中运用的更加广泛。

- Spring作为一个轻量级的框架，有很多的拓展功能，最主要的我们一般项目使用的就是IOC和AOP。
- SpringMVC是Spring实现的一个Web层，相当于Struts的框架，但是比Struts更加灵活和强大!
- Mybatis是 一个持久层的框架，在使用上相比Hibernate更加灵活，可以控制sql的编写，使用 XML或注解进行相关的配置!

根据上面的描述，学习SSM框架就非常的重要了!

二：搭建一个SSM的过程

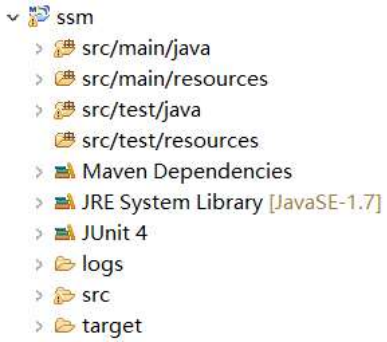
1. 使用Maven管理项目
   使用Maven在Eclipse中创建一个webapp的项目 ，具体的创建过程不做演示，如有不会创建的[创建项目]
   也可以使用Maven命令进行创建，在Dos窗口进入指定的目录，执行下面命令：

```
mvn archetype:create -DgroupId=org.ssm.dufy -DartifactId=ssm-demo -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
```

使用命令要注意，系统安装了Maven，并配置好了环境变量！[Maven的安装和环境变量配置]

1. 导入项目（命名创建），添加依赖
   导入项目是IDE中，或者直接在IDE创建，一般默认有【src/main/java】，手动创建【src/test/resources】、【src/test/java】文件夹。

   如下项目结构：



   然后直接配置 pom.xml文件中的包依赖!

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.dufy</groupId>
  <artifactId>ssm</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>ssmDemo</name>
  <url>http://maven.apache.org</url>
  <properties>
    <spring.version>4.0.5.RELEASE</spring.version>
    <mybatis.version>3.2.1</mybatis.version>
    <slf4j.version>1.6.6</slf4j.version>
    <log4j.version>1.2.12</log4j.version>
    <mysql.version>5.1.35</mysql.version>


  </properties>
  <dependencies>
  <!-- 添加Spring依赖 -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
```

shell编程(5)
Quartz系列(4)
Map(4)
Vector(4)
集合(4)
TreeMap(3)
更多

随笔分类

Java(20)
Java Web(5)
Linux学习(6)
Mybatis(1)
Mysql(3)
Nginx(1)
Quartz(8)
Redis(3)
Spring&SpringMVC(10)
阿飞的世界(2)
分布式相关(11)
开发工具(3)
算法

随笔档案

2018年7月 (2)
2018年6月 (1)
2018年3月 (8)
2018年1月 (2)
2017年10月 (3)
2017年9月 (1)
2017年8月 (1)
2017年7月 (10)
2017年6月 (9)
2017年4月 (7)
2017年3月 (16)
2017年2月 (12)

积分与排名

积分 - 27261
排名 - 16137

最新评论

1. Re:Spring-Session实现Session共享实现原理以及源码解析
@CodeLife-Jerry很高兴帮助到你！...
--阿飞(dufyun)
2. Re:Spring-Session实现Session共享实现原理以及源码解析
握草，作为新手的我查了这么久，你这一篇完全解决了我的疑问！赞！
--CodeLife-Jerry
3. Re:Java死锁排查和Java CPU 100% 排查的步骤整理
@博客园团队谢谢！...
--阿飞-dufyun
4. Re:Java死锁排查和Java CPU 100% 排查的步骤整理
您好，文中的代码块不能高亮显示，在代码块的```之外添加空行可解决
--博客园团队
5. Re:SSM框架——Spring+SpringMVC+Mybati的搭建教程
@阿飞-dufyunentity,controller,service这几个包的扫描都要在applicationContext.xml里配置吗...
--Youyou_2608

阅读排行榜

1. SSM框架——Spring+SpringMVC+Mybati的搭建教程(77613)

```
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context-support</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aspects</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <!--spring单元测试依赖 -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>${spring.version}</version>
            <scope>test</scope>
        </dependency>

<!-- spring webmvc相关jar -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>${spring.version}</version>
        </dependency>

<!-- mysql驱动包 -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>${mysql.version}</version>
    </dependency>

    <!-- alibaba data source 相关jar包-->
     <dependency>
         <groupId>com.alibaba</groupId>
         <artifactId>druid</artifactId>
         <version>0.2.23</version>
     </dependency>

     <!-- alibaba fastjson 格式化对 -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.1.41</version>
        </dependency>

     <!-- logback start -->
    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>${log4j.version}</version>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>${slf4j.version}</version>
    </dependency>
    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-classic</artifactId>
```

2. 《阿里巴巴Java开发手册》扫描插件正式发布--插件安装和使用分析(4591)
3. Quartz学习——Spring和Quartz集成详解（三）(2887)
4. Quartz学习——SSMM(Spring+SpringMVC+和Quartz集成详解（四）(23
5. Quartz学习——Quartz大致介绍（一）(1991)

评论排行榜

1. SSM框架——Spring+SpringMVC+Mybati的搭建教程(14)
2. 《阿里巴巴Java开发手册》扫描插件正式发布--插件安装和使用分析(8)
3. 一张图讲解对象锁和关键字synchronized修饰方法(3)
4. Java死锁排查和Java CPU 100% 排查的步骤整理(2)
5. Spring-Session实现Session共享实现原理以及源码解析(2)

推荐排行榜

1. SSM框架——Spring+SpringMVC+Mybati的搭建教程(2)
2. Quartz学习——SSMM(Spring+SpringMVC+和Quartz集成详解（四）(2)
3. Quartz学习——Quartz大致介绍（一）(2)
4. 《阿里巴巴Java开发手册》扫描插件正式发布--插件安装和使用分析(1)
5. Java死锁排查和Java CPU 100% 排查的步骤整理(1)

```
        <version>1.1.2</version>
    </dependency>
    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-core</artifactId>
        <version>1.1.2</version>
    </dependency>
    <dependency>
        <groupId>org.logback-extensions</groupId>
        <artifactId>logback-ext-spring</artifactId>
        <version>0.1.1</version>
    </dependency>

    <!--mybatis依赖 -->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>${mybatis.version}</version>
    </dependency>

    <!-- mybatis/spring包 -->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>1.2.0</version>
    </dependency>
  <!-- 添加servlet3.0核心包 -->
         <dependency>
             <groupId>javax.servlet</groupId>
             <artifactId>javax.servlet-api</artifactId>
             <version>3.0.1</version>
         </dependency>
         <dependency>
             <groupId>javax.servlet.jsp</groupId>
             <artifactId>javax.servlet.jsp-api</artifactId>
             <version>2.3.2-b01</version>
         </dependency>
         <!-- jstl -->
         <dependency>
             <groupId>javax.servlet</groupId>
             <artifactId>jstl</artifactId>
             <version>1.2</version>
         </dependency>
    <!--单元测试依赖 -->
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <finalName>ssmDemo</finalName>
  </build>
</project>
```

1. 创建数据库和表，生成代码

   创建数据库我参考别人的博客数据库设计，这块没有自己去书写，直接添上代码

```
DROP TABLE IF EXISTS `user_t`;

CREATE TABLE `user_t` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_name` varchar(40) NOT NULL,
  `password` varchar(255) NOT NULL,
  `age` int(4) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

/*Data for the table `user_t` */

insert  into `user_t`(`id`,`user_name`,`password`,`age`) values (1,'测试','sfasgfaf',24)
```

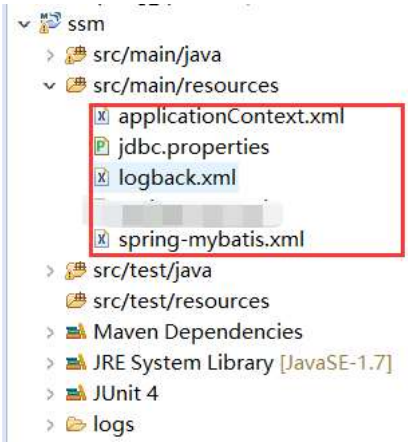
生成代码请查看：

[Mybatis自动生成代码]

生成的代码导入图片解释：

1. Spring 和 mybatis整合，连接数据库,进行Junit测试

   将生成的代码拷贝到项目中，然后进行Spring和Mybatis的整合，添加配置文件！



主要有

applicationContent.xml： Spring的相关配置！

Spring-mhbatis.xml： Spring和Mybatis集成配置！

jdbc.properties：数据库信息配置！

logback.xml：日志输出信息配置！（**不做介绍，详细信息查看源码**）

主要介绍applicationContext.xml、Spring-mhbatis.xml、jdbc.properties。主要内容如下：

jdbc.properties

```
jdbc_driverClassName =com.mysql.jdbc.Driver
jdbc_url=jdbc:mysql://localhost:3306/ssm?useUnicode=true&characterEncoding=utf8
jdbc_username=root
jdbc_password=root
```

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-
context-3.0.xsd
http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-
3.0.xsd
http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
">
    <!-- 1.配置jdbc文件 -->
    <bean id="propertyConfigurer"
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
            <property name="locations" value="classpath:jdbc.properties"/>
    </bean>

    <!-- 2.扫描的包路径，这里不扫描被@Controller注解的类 --><!--使用<context:component-scan/> 可以不在配置
<context:annotation-config/> -->
    <context:component-scan base-package="org.ssm.dufy">
        <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

    <import resource="classpath:spring-mybatis.xml" />
```

```
</beans>
```

spring-mybatis.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
   http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-
3.2.xsd
   http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-
3.2.xsd
   http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd">

<!-- 3.配置数据源 , 使用的alibba的数据库-->
     <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-
method="close">
        <!-- 基本属性 url、user、password -->
        <property name="driverClassName" value="${jdbc_driverClassName}"/>
        <property name="url" value="${jdbc_url}"/>
        <property name="username" value="${jdbc_username}"/>
        <property name="password" value="${jdbc_password}"/>

        <!-- 配置初始化大小、最小、最大 -->
        <property name="initialSize" value="10"/>
        <property name="minIdle" value="10"/>
        <property name="maxActive" value="50"/>

        <!-- 配置获取连接等待超时的时间 -->
        <property name="maxWait" value="60000"/>
        <!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->
        <property name="timeBetweenEvictionRunsMillis" value="60000" />

        <!-- 配置一个连接在池中最小生存的时间，单位是毫秒 -->
        <property name="minEvictableIdleTimeMillis" value="300000" />

        <property name="validationQuery" value="SELECT 'x'" />
        <property name="testWhileIdle" value="true" />
        <property name="testOnBorrow" value="false" />
        <property name="testOnReturn" value="false" />

        <!-- 打开PSCache，并且指定每个连接上PSCache的大小  如果用Oracle，则把poolPreparedStatements配置为
true，mysql可以配置为false。-->
        <property name="poolPreparedStatements" value="false" />
        <property name="maxPoolPreparedStatementPerConnectionSize" value="20" />

        <!-- 配置监控统计拦截的filters -->
        <property name="filters" value="wall,stat" />
    </bean>



    <!-- spring和MyBatis完美整合，不需要mybatis的配置映射文件 -->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <!-- 自动扫描mapping.xml文件 -->
        <property name="mapperLocations" value="classpath:org/ssm/dufy/mapper/*.xml"></property>
    </bean>


     <!-- DAO接口所在包名，Spring会自动查找其下的类 ,自动扫描了所有的XxxxMapper.xml对应的mapper接口文件,只要
Mapper接口类和Mapper映射文件对应起来就可以了-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="org.ssm.dufy.dao" />
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"></property>
    </bean>

    <!-- (事务管理)transaction manager, use JtaTransactionManager for global tx -->
<!-- 配置事务管理器 -->
    <bean id="transactionManager"
        class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSource" />
    </bean>

    <!--======== 事务配置 End ================== -->
    <!-- 配置基于注解的声明式事务 -->
    <!-- enables scanning for @Transactional annotations -->
    <tx:annotation-driven transaction-manager="transactionManager" />
```

```
</beans>
```

测试代码，两种方式：

测试1：

```
package org.ssm.dufy.service;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.ssm.dufy.entity.User;

/**
 * 配置spring和junit整合，junit启动时加载springIOC容器 spring-test,junit
 */
@RunWith(SpringJUnit4ClassRunner.class)
// 告诉junit spring配置文件
@ContextConfiguration({ "classpath:applicationContext.xml"})
public class IUserServiceTest {

    @Autowired
    public IUserService userService;

    @Test
    public void getUserByIdTest(){

        User user = userService.getUserById(1);
        System.out.println(user.getUserName());
    }

}
```

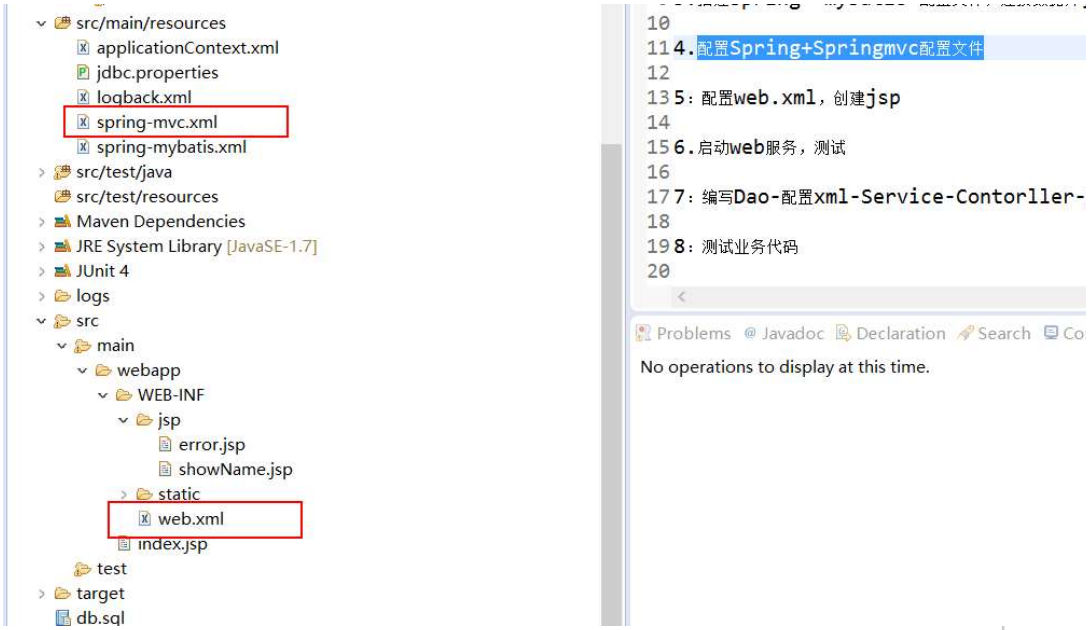
测试2：

```
package org.ssm.dufy.service;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.ssm.dufy.entity.User;

public class IUserServiceTest2 {


    public static void main(String[] args) {
        ApplicationContext application = new
ClassPathXmlApplicationContext("applicationContext.xml");
        IUserService uService = (IUserService) application.getBean("userService");
        User user = uService.getUserById(1);
        System.out.println(user.getUserName());
    }
}
```

5：整合SpringMVC，添加配置，创建jsp

SpringMVC需要的依赖在pom.xml中已经加上了，现在需在Web项目中的web.xml中添加启动SpringMVC启动配置和Spring整合SpringMVC的配置了。

新增如下两个文件：



spring-mvc.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:p="http://www.springframework.org/schema/p"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.2.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">

    <!-- 扫描controller（controller层注入） -->
   <context:component-scan base-package="org.ssm.dufy.web" use-default-filters="false">
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

     <mvc:annotation-driven />

      <!-- 内容协商管理器  -->
    <!--1、首先检查路径扩展名（如my.pdf）；2、其次检查Parameter（如my?format=pdf）；3、检查Accept Header-->
    <bean id="contentNegotiationManager"
class="org.springframework.web.accept.ContentNegotiationManagerFactoryBean">
        <!-- 扩展名至mimeType的映射,即 /user.json => application/json -->
        <property name="favorPathExtension" value="true"/>
        <!-- 用于开启 /userinfo/123?format=json 的支持 -->
        <property name="favorParameter" value="true"/>
        <property name="parameterName" value="format"/>
        <!-- 是否忽略Accept Header -->
        <property name="ignoreAcceptHeader" value="false"/>

        <property name="mediaTypes"> <!--扩展名到MIME的映射；favorPathExtension, favorParameter是true时
起作用  -->
            <value>
                json=application/json
                xml=application/xml
                html=text/html
            </value>
        </property>
        <!-- 默认的content type -->
        <property name="defaultContentType" value="text/html"/>
    </bean>


    <!-- 当在web.xml 中   DispatcherServlet使用 <url-pattern>/</url-pattern> 映射时，能映射静态资源 -->
    <mvc:default-servlet-handler />
    <!-- 静态资源映射 -->
    <mvc:resources mapping="/static/**" location="/WEB-INF/static/"/>


   <!-- 对模型视图添加前后缀 -->
     <bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver"
      p:prefix="/WEB-INF/jsp/" p:suffix=".jsp"/>
```

```
</beans>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <display-name>SSM-DEMO</display-name>
  <!-- 读取spring配置文件 -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>

    <!-- 设计路径变量值
    <context-param>
        <param-name>webAppRootKey</param-name>
        <param-value>springmvc.root</param-value>
    </context-param>
    -->

    <!-- Spring字符集过滤器 -->
    <filter>
        <filter-name>SpringEncodingFilter</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
        <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
        <init-param>
            <param-name>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>SpringEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!-- 添加日志监听器 -->
    <context-param>
        <param-name>logbackConfigLocation</param-name>
        <param-value>classpath:logback.xml</param-value>
    </context-param>
    <listener>
        <listener-class>ch.qos.logback.ext.spring.web.LogbackConfigListener</listener-class>
    </listener>

    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <!-- springMVC核心配置 -->
    <servlet>
        <servlet-name>dispatcherServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <!--spingMVC的配置路径 -->
            <param-value>classpath:spring-mvc.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <!-- 拦截设置 -->
    <servlet-mapping>
        <servlet-name>dispatcherServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>


  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

新增index.jsp文件

```
<%@ page contentType="text/html; charset=utf-8"%>
<!doctype html>
<html>
<body>
<h2>Hello World!</h2>
</body>
</html>
```

6.启动web服务，测试

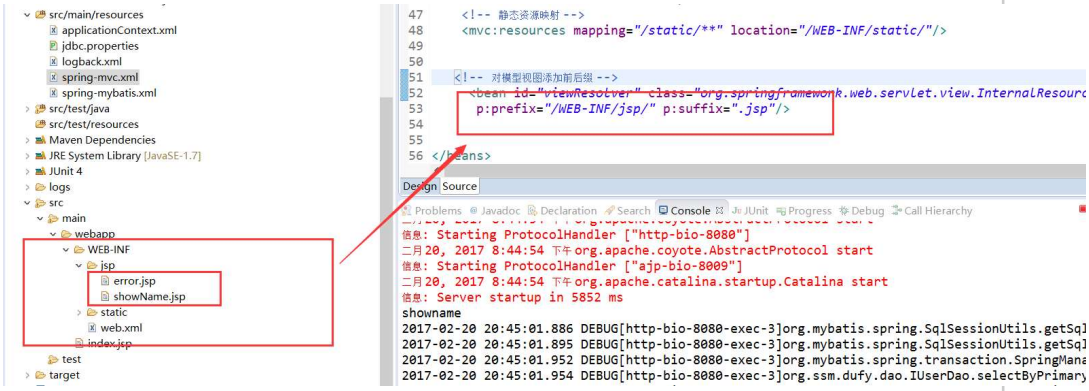
将上面的项目添加到Tomcat中，启动，控制台没有报错，并在地址栏访问，http://localhost:8080/ssm。页面显示Hello World! 项目配置ok!

7：编写Controller，和对应得业务界面

新增UserController ，通过参数传递uid获取用户，若用户存在，跳转到showName.jsp ，若用户不存在，则跳转到error.jsp,并返回提示信息!

```
package org.ssm.dufy.web;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;
import org.ssm.dufy.entity.User;
import org.ssm.dufy.service.IUserService;

@Controller
public class UserController {

    @Autowired
    private IUserService userService;

    @RequestMapping(value="/showname",method=RequestMethod.GET)
    public String showUserName(@RequestParam("uid") int uid,HttpServletRequest request,Model model){
        System.out.println("showname");
        User user = userService.getUserById(uid);
        if(user != null){
            request.setAttribute("name", user.getUserName());
            model.addAttribute("mame", user.getUserName());
            return "showName";
        }
        request.setAttribute("error", "没有找到该用户！");
        return "error";
    }
}
```

Jsp内容如下：



showName.jsp

```
<%@ page contentType="text/html; charset=utf-8"%>
<!doctype html>
<html>
<head>
    <title>show name</title>
</head>
<body>
    <h1>Welcome</h1> ${name }<h1>访问此页面</h1>

</body>
</html>
```

error.jsp

```
<%@ page contentType="text/html; charset=utf-8"%>
<!doctype html>
<html>
<head>
    <title>error page</title>
</head>
<body>
    <h2> ${error } </h2>
```

```
</body>
</html>
```

三：遇到的问题

1:找不到UserService，报错

> 可能是包扫描路径的问题，检查一下Service是否在扫描的范围内

2：jsp接收不到request.setAttribute("","");内容

> 查资料说是因为 JSP的版本问题，可以在Jsp 上面添加 <%@ page isELIgnored="false"
> %>
> 或者修改web.xml ,添加version版本!

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd"
    id="WebApp_ID" version="3.0" metadata-complete="true">
```

四：心得总结

或许这些都是站在别人的基础的搭建的，但是看一篇SSM的搭建文章可能很快，看完觉得自己懂了，但是我建议一定要自己去搭建一次，看一遍，和动手做一遍完全是两个概念!

五：参考文章

SSM框架——详细整合教程（Spring+SpringMVC+MyBatis）

Mybatis3.x与Spring4.x整合

附:

项目源代码:

https://github.com/dufyun/learn-tech-collection/tree/master/ssm

---

欢迎访问我的csdn博客，我们一同成长!

"不管做什么，只要坚持下去就会看到不一样！在路上，不卑不亢!"

http://blog.csdn.net/u010648555

分类: Spring&SpringMVC

标签: SSM框架, SpringMVC, Spring, Mybatis

阿飞(dufyun)
关注 - 3
粉丝 - 29
+加关注

2 1

« 上一篇：Navicat for MySQL：快捷键整理
» 下一篇：java集合系列——java集合概述（一）

posted on 2017-02-20 20:58 阿飞(dufyun) 阅读(77613) 评论(14) 编辑 收藏

评论

#1楼

哥，你依赖里面两个spring-webmvc，maven能自动下载才怪你，你坑人啊

支持(0) 反对(0)

2017-12-17 16:32 | LAOK23

#2楼[楼主]